สนุกกับแอสเซมบลี้ ตอน 3

ADDRESSING MODE

กฤษดา กิติสาระกุลชัย

สวัสดีครับท่านผู้อ่านที่รักทั้งหลาย เหม! ผมหายไปตั้งนานคิดถึงคุณจังเลย กลับมาคราวนี้ผมต้องขอโทษทุก ๆ คนก่อนเลย ยิ่งโดยเฉพาะคนที่รอบทความนี้อยู่ ผมขอรับผิดทุกอย่างเลยที่ทำให้บทความนี้หายไป และต้องขอขอบคุณผู้ที่จดหมายมาถึงด้วยครับ ก็ขอเริ่มกันเลยดีกว่าครับ เพราะคราวนี้เป็นเรื่องใหญ่ซะด้วยซิครับ คราวนี้ผมจะกล่าวถึงเรื่องสำคัญเรื่องหนึ่งคือเรื่องแอดเดรสซิ่งโหมด (Addressing Mode) กับเรื่องชุดคำสั่งการย้ายข้อมูลอีกเรื่องหนึ่ง ที่มาดูกันละครับว่าที่ว่าเรื่องใหญ่นะมันใหญ่ขนาดไหน

ก่อนอื่นก็ต้องขอสารภาพไว้ซะตรงนี้เลยว่า การเขียนบทความในตอนนี้มันยากเสียเหลือเกิน ทั้งนี้เพราะว่า เรื่องชุดคำสั่งที่จะพูดออกไปนั้น มันจะต้องเกี่ยวข้องกับการใช้แอดเดรสซิ่ง (addressing) ตั้งหลายแบบ แต่ครั้นจะพูดถึงเรื่องแอดเดรสซิ่งก่อน คุณก็จะต้องเข้าใจเรื่องการใช้คำสั่งซะก่อน เลย ไม่รู้ว่าจะทำยังไงดี คุณถึงจะเข้าใจได้ทั้งสองอย่างพร้อม ๆ กัน ก็เลยลองเขียนมันขึ้นมาทั้งสองแบบเลย แล้วไปให้เพื่อนสองคนอ่านคนละฉบับ เพื่อดูว่าใครจะมีปัญหาบ้าง ปรากฏว่ามีปัญหาทั้งสองคนเลย แต่มีปัญหาต่างกันออกไป สุดท้ายก็เลยสรุปว่า จะพูดเรื่องแอดเดรสซิ่งโหมดก่อน โดย (ไมเม) คิดว่าคุณเข้าใจการใช้คำสั่งแล้ว ดังนั้นหากคุณอ่านบทความนี้แล้วยังไม่เข้าใจ คุณก็จะต้องทำในทางกลับกันด้วยนะครับ คือ อ่านเรื่องแอดเดรสซิ่งแล้วไปอ่านเรื่องชุดคำสั่ง แล้วกลับมาแล ๆ เรื่องแอดเดรสซิ่งอีกซักกะที

addressing mode

อ่านว่า "แอดเดรสซิ่ง โหมด" ชื่อแปลกดีนะครับ คล้ายกับรถซิ่งเลยนะ จะชื่อจะแซ่อะไรไม่สำคัญเท่ากับว่า มันมีตั้ง 13 โหมดแน่ ถ้าขืนไม่ยอม "ซิ่ง" ซะตอนนี้เดี๋ยวจะไม่จบในเล่มเอานะครับ ก็ว่ากันไปเลยว่าทั้ง 13 โหมด มีดังนี้

1. Immediate Addressing Mode

คำสั่งที่ใช้โหมดนี้ จะต้องการหน่วยความจำ 2 ไบท์ คือไบท์หนึ่งของออปโค้ดกับอีกไบท์หนึ่งสำหรับโอเปอร์แรนท์ (operand) โอเปอร์แรนท์ของโหมดนี้จะเป็นไบท์ที่สองของคำสั่ง

ตัวอย่างเช่น

```
7000-  A9 09      LDA  #$09
```

ก็หมายความว่า ให้เอาค่า $09 จากไบท์ที่สองของคำสั่ง (LDA) ไปกระทำ (ในที่นี้คือ เอาไปเก็บไว้ในแอคคิวมูเลเตอร์)

2. Absolute Addressing Mode

โหมดนี้ ไบท์ที่สองกับไบท์ที่สามของคำสั่งจะเป็นแอดเดรสของโอเปอร์แรนท์ โดยที่ไบท์ที่สามจะบอกหน้า (page) ของหน่วยความจำ และไบท์ที่สองจะบอกตำแหน่งในหน้า (page) นั้น หรือพูดอีกอย่างก็คือ ไบท์ที่สองจะเป็น ADL (แอดเดรสค่าน้อย หรือสองตำแหน่งหลังของแอดเดรส) และไบท์ที่สามเป็น ADH (แอดเดรสค่ามาก หรือสองตำแหน่งหน้าของแอดเดรส) เช่น ถ้าไบท์ที่สองเป็น $00 และไบท์ที่สามเป็น $C0 ค่า ADL จะเป็น $00 และ ADH จะเป็น $C0 จะได้ว่าโอเปอร์แรนท์เป็น $C000

ตัวอย่างเช่น

```
7000-  AD 00 C0   LDA  $C000
```

มีความหมายว่า ให้เอาค่าในแอดเดรส $C000 ใส่ในแอคคิวมูเลเตอร์

3. Zero Page Addressing Mode

โหมดนี้ ไบท์ที่สองของคำสั่งจะเป็นแอดเดรสของโอเปอร์แรนท์ ซึ่งอยู่ในหน้าที่ $00 (Zero Page) คือ ADL คือไบท์ที่สอง และ ADH มีค่าเท่ากับ $00 ซึ่งจะมีลักษณะคล้ายกับ Absolute Addressing Mode มาก

ตัวอย่างเช่น

```
7000-   A5 67        LDA   $67
```

หากจะใช้ Absolute Addressing Mode ก็จะแทนที่ลงไปได้เลยว่า

```
7000-   AD 67 00     LDA   $0067
```

จะต่างกันก็คือ Absolute Addressing Mode จะเปลืองหน่วยความจำมากกว่าอยู่ 1 ไบท์

4. Relative Addressing Mode

การแอดเดรสซิ่งแบบนี้มีใช้เฉพาะในคำสั่งที่เกี่ยวกับบรานซ์ (branch) เท่านั้นเอง ถ้าเงื่อนไขในคำสั่งบรานซ์ถูกต้อง จะมีการบรานซ์เกิดขึ้น ซึ่งแอดเดรสที่จะบรานซ์ไปถึงนั้นจะเป็นแอดเดรสที่เกิดจากค่าในไบท์ที่สองของคำสั่ง บวกเข้ากับแอดเดรสที่ต่อจากคำสั่งบรานซ์นั้น ค่าในไบท์ที่สองนี้มีทั้งค่าบวกและค่าลบด้วยนะครับ การกำหนดว่าเป็นค่าบวกหรือลบจะใช้บิทที่อยู่ซ้ายสุดหรือบิทที่ 7 หรือบิทที่มีค่ามากที่สุดเป็นตัวกำหนด คือ ถ้ามีค่าเป็น 0 ถือเป็นบวก ถ้ามีค่าเป็น 1 ถือเป็นลบ ซึ่งจะทำให้เกิดการบรานซ์ได้ทั้ง 2 ทาง คือทางบวกหรือไปข้างหน้าหรือไปที่แอดเดรสมากขึ้น กับอีกทางหนึ่งคือเป็นลบหรือย้อนกลับหรือไปที่แอดเดรสที่น้อยลง ซึ่งจะทำให้ระยะทางที่บรานซ์ไปนั้นไปได้ไกลเพียง 127 ไบท์เท่านั้น ถ้าต้องการบรานซ์ให้ไกลกว่านี้จะต้องใช้เทคนิคอื่นช่วย เช่น การบรานซ์ซ้อนหรือการ JMP

ตัวอย่างเช่น

```
6000-   F0 07        BEQ   $6009
```

ไบท์ที่สองเป็น $07 มีค่าเป็นบวก แอดเดรสที่ต่อจากคำสั่งนี้คือ $6002 ก็จะบรานซ์ไปที่ $6002 + $07 = $6009 หรือ

```
6000-   F0 F9        BEQ   $5FFB
```

ไบท์ที่สองเป็น $F9 ซึ่งบิทที่ 7 มีค่าเป็น 1 จึงทำให้มีค่าเท่ากับ $-07 ก็จะบรานซ์ไปที่ $6002 + ($-07) = $5FFB

5. Accumulator Addressing Mode

โหมดนี้โอเปอร์แรนท์จะเป็นแอคคิวมูเลเตอร์เองเลย คำสั่งในแอดเดรสซิ่งโหมดนี้ ต้องการหน่วยความจำเพียงไบท์เดียว

ตัวอย่างเช่น

```
6000-   6A           ROR
```

ก็หมายความว่า การกระทำ ROR นี้จะเกิดขึ้นกับแอคคิวมูเลเตอร์

6. Implied Addressing Mode

โหมดนี้โอเปอร์แรนท์จะขึ้นอยู่กับคำสั่งที่ใช้ ส่วนใหญ่จะเป็นรีจิสเตอร์ใน 6502 เอง หรือไม่ก็จะอยู่ในสแต็ก (stack) (อยู่ที่เพจ 1 ของหน่วยความจำ) คำสั่งที่ใช้ในโหมดนี้ ใช้หน่วยความจำเพียงไบท์เดียว

ตัวอย่างเช่น

INX โอเปอร์แรนท์ คือ รีจิสเตอร์ X
INY โอเปอร์แรนท์ คือ รีจิสเตอร์ Y
RTS โอเปอร์แรนท์ คือ ค่าในสแต็ก

7. Absolute Indexed by X Addressing Mode

โหมดนี้โอเปอร์แรนท์จะอยู่ที่หน่วยความจำ [ADH-ADL] โดยที่ ADL จะเท่ากับผลบวกค่าในรีจิสเตอร์ X กับค่าในไบท์ที่สองของคำสั่ง ส่วน ADH นั้นจะเป็นไบท์ที่สาม บวกกับตัวทด (carry) ที่อาจจะเกิดขึ้นเนื่องจากการบวกเพื่อหา ADL ถ้าผลบวกเพื่อหา ADL มีค่าเกิน $FF จะทำให้เกิดมีตัวทดขึ้น ก็จะเอาตัวทดซึ่งมีค่าเป็น 1 ไปบวกกับไบท์ที่สามได้เป็น ADH แต่ถ้าผลบวกไม่เกิน $FF ตัวทดก็จะไม่มีหรือมีค่าเป็น 0 ADH ก็จะมีค่าเท่ากับไบท์ที่สามของคำสั่ง

ตัวอย่างเช่น

```
6000-   9D FF 01     STA   $01FF,X
```

ถ้าค่า X เป็น $00

ADL = $FF + $00
= $FF ไม่เกิดตัวทด
ADH = $01 + c = $01 c = 0
Operand = $01FF

ถ้าค่าใน X เป็น $58

ADL = $FF + $58
= $157 ซึ่งค่าเกิน $FF ทำให้มีตัวท
เกิดขึ้นมาทำให้
ADL = $57 c = 1
ADH = $01 + c = $02
Operand = $0257

8. Absolute Indexed by Y Addressi Mode

โหมดนี้จะเหมือนแอดเดรสซิ่งโหมด 7 เกือบเปี๊ยบเลย ต่างกันตรงที่เอาค่า Y เป็นอินเด็กซ์ แทนที่จะเป็น X เท่านั้นเอง

9. Zero Page Indexed by X Addressi Mode

โหมดนี้จะคล้ายกับแอดเดรสซิ่งโห ที่ 7 แต่ก็มีที่แตกต่างอยู่ 2 ที่ คือ

1. โหมดนี้ใช้หน่วยความจำ 2 ไ ก็พอแทนที่จะเป็น 3 ไบท์

2. การหา ADL ได้จากการบวกไ ที่สองของคำสั่งกับค่าในรีจิสเตอร์ X แต่ ไม่คำนึงถึงการเกิดตัวทดขึ้นเลย ถ้าค่าที่บ ได้เกิน $FF ก็จะเอาสองตำแหน่งหลังเ ADL ซะเฉย ๆ เลย ดังนั้นค่า ADH ก็ มีค่าเป็น $00 ลูกเดียวเลย

ตัวอย่างเช่น

```
6000-   94 F0        STY   $F0,X
```

หมายความว่า ให้เอาค่าในรีจิสเตอร์ไปเ ไว้ที่แอดเดรส $00F0 + X
สมมติว่า X มีค่าเป็น $09 แอดเดรสที่จะค่าใน Y ไปเก็บ คือ

$00F0 + $09 = $00F9

แต่ถ้า X มีค่าเป็น $11 ผลบวกของไบ สองกับค่าใน X จะเป็น

$F0 + $11 = $101

ือตัดตัวทดออก ค่า ADL จะเป็น $01 ว่าค่า ADH ยังเป็น $00 อยู่ดี

ดังนั้นโอเปอร์แรนท์ หรือแอดเดรสที่ะเอาค่าใน Y ไปเก็บก็คือ $0001 แทนที่ะเป็น $0101 ถ้าใช้แอดเดรสซึ่งในโหมดที่ 7

0. Zero Page Indexed by Y Addressing Mode

โหมดนี้ก็เหมือนกับแอดเดรสซึ่งโหมด 9 นั่นเอง ต่างกันตรงที่รีจิสเตอร์ Y แทนที่ะเป็นรีจิสเตอร์ X เท่านั้นเอง

1. Indirect Addressing Mode

โหมดนี้จะมีใช้กับคำสั่ง JMP เพียงสั่งเดียวเท่านั้นเอง ไบท์ที่สองและไบท์สามของคำสั่งนี้ จะเป็น ADL และ ADH องแอดเดรสที่จะ JMP ไป คุณคงจะพอได้ว่าคำสั่ง JMP นั้นจะกระโดดไปยัง CH-PCL ใหม่ โดยไม่มีการเก็บค่าใด ๆ ในสแต็กเลย ถ้าเป็น Absolute Addressing Mode ตามที่ผมได้เคยเขียนไปแล้ว CH-PCL ใหม่ก็จะเป็นไบท์ที่สองกับไบท์สามของคำสั่งนี้ ซึ่งจะตามหลังออปโค้ด (JMP) เลย แต่สำหรับ JMP ในโหมดนี้ ไบท์ที่สองกับไบท์ที่สามของคำสั่งจะเป็นตัวอกตำแหน่งของ PCH-PCL ใหม่ ให้ว่าที่ไหน โดย

PCL ใหม่ = [ADH−ADL]
PCH ใหม่ = [ADH−ADL+1]

ตัวอย่างเช่น

```
FDED-   6C 36 00    JMP   ($0036)
```

หมายความว่า ให้ JMP ไปที่แอดเดรสที่หนดโดยแอดเดรสที่ $0036 กับ $0037 ไปดูที่ $0036 กับ $0037 แล้วพบว่ามีค่า 0 กับ $FD ก็หมายความว่า ค่า PCL ใหม่เป็น $F0 และ PCH ใหม่ก็จะเป็น $FD สั่ง JMP นี้ก็จะ JMP ไปที่ $FDF0

คำสั่ง JMP ที่ใช้โหมดนี้ และที่ตำแหน่งนี้ ($FDED) แอบเปิ้ลเอามาใช้เป็นท์พุทซีเล็ค (output select) เพื่อเลือกว่าจะให้เอาท์พุทออกทางเอาท์พุทชนิดใด เพราะที่ $FDED นี้เป็นตำแหน่งที่อยู่ในรอม ไม่สามารถเปลี่ยนได้ (ง่าย ๆ) วิธีที่ใช้ในการเปลี่ยนเอาท์พุทจึงใช้วิธีไปเปลี่ยนที่ตำแหน่ง $0036 และ $0037 แทน เพราะอยู่นอกรอม ดังนั้นถ้าคุณไม่ต้องการให้แอปเปิ้ลของคุณแขวนคอตาย (hang) ละก้อ คุณก็อย่าได้ไป (ซี้ซั้ว) เปลี่ยนค่าที่อยู่ในแอดเดรสทั้งสองนี้นะครับ

12. Indirect Indexed Addressing Mode

โหมดนี้จะมีลักษณะคล้ายกับโหมดที่ 11 แต่ชื่อก็บอกอยู่แล้วว่าเกี่ยวกับอินเด็กซ์ด้วย และอินเด็กซ์ที่ใช้ในโหมดนี้ก็จำกัดอยู่ให้เป็นรีจิสเตอร์ Y เท่านั้น โหมดนี้ต้องการหน่วยความจำ 2 ไบท์ ไบท์หนึ่งสำหรับออปโค้ด อีกไบท์หนึ่งจะเป็น Indirect Address Low ในที่นี้จะขอย่อว่า IAL ส่วน IAH ก็ถูกจำกัดว่าเป็น $00 เท่านั้นอีก จึงหมายความว่า Indirect Address นี้จะเป็นแอดเดรสใน zero page เท่านั้น ซึ่งต่างจาก Indirect Addressing Mode ซึ่งไม่จำกัดว่าจะต้องเป็น zero page การหาโอเปอร์แรนท์ของโหมดนี้ก็หาได้ดังนี้

ADL = [IAL] + [Y]
ADH = [IAL+1] + C
Operand = [ADH−ADL]

ซึ่งการบวกในการหา ADL นั้น จะต้องคำนึงถึงการเกิดตัวทดด้วย ลักษณะจะคล้ายกับในโหมดที่ 8 ที่มีไบท์ที่สองเป็น IAL และไบท์ที่สามเป็น IAL+1

ตัวอย่างเช่น

```
6000-   B1 67       LDA   ($67),Y
```

ถ้าค่าที่แอดเดรส $67 กับ $68 เป็น $01 กับ $08 ตามลำดับ และค่าใน Y เป็น $02 แล้ว โอเปอร์แรนท์จะเป็นแอดเดรส

$0801+$02 = $0803

แต่ถ้า Y เป็น $FF แล้ว โอเปอร์แรนท์จะเป็นแอดเดรส

$0801+$FF = $0900

ถ้าจะเขียนโดยใช้โหมดที่ 8 ก็จะได้เป็น

```
6000-   B9 01 08    LDA   $0801,Y
```

13. Indexed Indirect Addressing Mode

เพื่อไม่ให้อินเด็กซ์น้อยหน้ากัน โหมดนี้จึงใช้รีจิสเตอร์ X เท่านั้นเป็นอินเด็กซ์ ลักษณะก็จะคล้ายกับ Indirect Addressing Mode เหมือนกัน แต่ต่างกันตรงที่โหมดนี้จะเอาอินเด็กซ์ (รีจิสเตอร์ X) บวกกับ IAL ก่อน ค่าที่ได้จะเป็น ADL เลย และไม่คำนึงถึงตัวทดด้วย ส่วน ADH ก็จะเป็นแอดเดรสถัดกับ ADL ไป การหาโอเปอร์แรนท์จึงพอสรุปได้ดังนี้

ADL = [IAL+[X]]
ADH = [IAL+[X]+1]
Operand = [ADH−ADL]

การยกตัวอย่างในโหมดนี้ ผมจะไม่ยกละครับ ให้คุณลองเอาเองจากเครื่องเลย คุณจะเข้าใจได้ดีขึ้น โปรแกรมที่ผมใช้ทดลองก็เป็นดังรูปที่ 1 ครับ

```
6000-   A6 00       LDX   $00
6002-   A1 80       LDA   ($80,X)
6004-   85 01       STA   $01
6006-   60          RTS
```

1

ก่อนที่คุณจะ G(RUN) โปรแกรมนี้ คุณก็ตรวจดูหน่วยความจำที่คุณสงสัยใน zero page ดูก่อนแล้วเปลี่ยนค่า x ในแอดเดรส $0000 ไปต่าง ๆ นานา พอ G เสร็จก็ดูว่าที่แอดเดรส $0001 เก็บค่าอะไรไว้ และค่านั้นมาจากแอดเดรสอะไร

นั่นก็เป็นเรื่องใหญ่ที่ผมพูดถึงละครับ สลบกันไปกี่คนแล้วก็ไม่รู้ อย่าพึ่งงงกันก่อนละครับ ผมก็บอกไปแล้วว่าไม่รู้ว่าจะทำยังไงคุณถึงจะไม่งง ก็ทิ้งไว้ก่อนแล้วกัน มาดูชุดคำสั่งกันอีกซักชุดดีกว่า เดี๋ยวคุณจะไม่หายกันไม้คันมือ ในการเขียนโปรแกรมกัน ขอให้นำตารางสรุปชุดคำสั่งออกมากางดูได้แล้วละครับ พร้อมหรือยังครับ

ชุดคำสั่งการย้ายข้อมูล

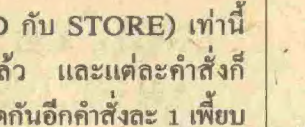
คำสั่งในตระกูลนี้หรือในชุดนี้ ก็มีเพียง LOAD กับ STORE เท่านั้นเองละครับ

แต่พอเห็นคำว่า "เพียง" กับ "เท่านั้น" แล้ว คิดว่าน้อย หามิได้ครับ เพราะถึงจะ "แค่" 2 อย่าง (คือ LOAD กับ STORE) เท่านี้ก็ปาเข้าไป 6 คำสั่งแล้ว และแต่ละคำสั่งก็ยังมีแอดเดรสซึ่งโหมดกันอีกคำสั่งละ 1 เพี้ยบ จนบางทีทำให้พอคุณอ่านจบ คุณอาจจะร้องออกมาว่า "เฮ้ย! จะรีบไปไหนกัน" และอาจจะสงกะสัยว่า ใช่คนเขียนคนเก่าหรือเปล่านะ เพราะบทความตอนนี้เขียนยังกับติดปีกไว้กับปากกาแน่ะ ก็ต้องขอตอบว่าคนเก่า คนเดิมครับ แต่ไม่ได้ติดปีกหรอกครับมันเกะกะ ทั้งนี้เพราะว่า ใจจริงแล้ว อยากจะให้บทความจบเป็นตอน ๆ จะได้ดูแล้วไม่ขาดไปห้วน ๆ ด้วน ๆ ฮู้ ๆ ...ในขณะเดียวกันก็อยากจะให้คุณเขียนโปรแกรมด้วยตัวเองมากขึ้น แต่พอมาเจอกับเรื่องแอดเดรสซิ่งโหมดเข้าให้ ก็เลยทำให้รู้สึกว่า จะต้องเก็บปีก แล้วมาติดเครื่องพลังเทอร์โบเจ็ทแทน ไม่งั้นไม่จบในเล่มแน่ หรือถ้าจบคุณก็คงไม่รู้จักคำสั่งเพิ่ม ก็เลยต้องขออนุญาตคุณละครับ และสงสัยว่า จะต้องติดอีกเครื่องหนึ่งเพื่อจะอธิบายคำสั่งในชุดนี้ให้จบในเล่มนี้ด้วย โดยผมจะรวมคำสั่งในชุดนี้เป็น 2 เทือก คือเทือกของการ LOAD กับ STORE

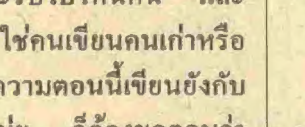
เทือกของ LOAD ก็มี LDA, LDX และ LDY อักษร LD นี้มาจาก LOAD ส่วนอักษร A, X หรือ Y ที่ตามหลังมา ก็คงจะพอเดากันได้ว่ามาจากแอคคิวมูเลเตอร์, รีจิสเตอร์ X และรีจิสเตอร์ Y ตามลำดับ พอรู้ที่มาของอักษรย่อแล้ว ก็คงจะพอเข้าใจคำสั่งได้บ้างแล้ว คือคำสั่งในเทือกนี้จะนำค่าในโอเปอร์แรนท์ (ซึ่งขึ้นอยู่กับการแอดแดรสซิ่ง) ไปเก็บไว้ในรีจิสเตอร์ซึ่งจะเป็นรีจิสเตอร์ตัวใด ก็ขึ้นอยู่กับคำสั่งว่าเป็น LDA, LDX หรือ LDY

เทือกของ STORE ก็มี STA, STX และ STY ก็เหมือนกันละครับ ST ก็มาจาก STORE คำสั่งในเทือกนี้จะกลับกับ LOAD คือจะเอาค่าในโอเปอร์แรนท์ไปเก็บไว้ในรีจิสเตอร์ ซึ่งก็เช่นกัน ขึ้นอยู่กับว่าเป็นคำสั่ง STA, STX หรือ STY

ตัวอย่างเช่น สมมติว่าที่แอดเดรส $00 เป็น $4C และแอคคิวมูเลเตอร์เป็น $96



ค่า $96 ในแอคคิวมูเลเตอร์จะถูกลบหายไป หรือถ้า



ค่า $4C ใน $00 ก็จะถูกลบหายไปเช่นกัน ลองมาดูโปรแกรมกันดีกว่านะครับ ดังในรูปที่ 2

```
7000-  A9 30     LDA  #$30
7002-  20 A8 FC  JSR  $FCA8
7005-  AD 30 C0  LDA  $C030
7008-  4C 00 70  JMP  $7000
```

2

โปรแกรมนี้จะทำให้ลำโพงเล็ก ๆ ของเครื่องคราง ออกมาจนกว่าคุณจะกดรีเซ็ท การทำงานของโปรแกรมนี้ก็คือ ซับรูทีนที่แอดเดรส $FCA8 ชื่อว่า WAIT โดยจะถ่วงเวลาในการไปทำคำสั่งอื่น เวลาที่ถ่วงเอาไว้จะขึ้นอยู่กับค่าในแอคคิวมูเลเตอร์ ดังนั้นจึงต้องกำหนดค่าในแอคคิวมูเลเตอร์ด้วยคำสั่ง LDA # $30 ค่า 30 นี้คุณสามารถเปลี่ยนเป็นค่าอื่น ๆ ได้ ก็จะทำให้ได้เสียงต่างออกไป ส่วนคำสั่ง LDA $C030 ก็จะทำให้ลำโพงกระเพื่อม 1 ครั้ง เมื่อกระเพื่อมหลายครั้งเข้าด้วยคำสั่ง JMP $7000 ใน 1 วินาที ก็จะทำให้เกิดเสียงที่มีความถี่ต่างกันออกไปขึ้นกับการหน่วงเวลาของคำสั่ง JSR $FCA8

โปรแกรมในรูปที่ 3 จะเป็นโปรแกรมเปลี่ยนเลขจากฐานสิบตั้งแต่ 0–65535 เป็นเลขฐานสิบหกจำนวน 4 หลัก คือ $0000-$FFFF โดยการอินพุทตัวเลขเข้าไป การทำงานของโปรแกรมเป็นดังนี้ครับ ซับรูทีน $D52E จะทำหน้าที่เปิดโอกาสให้คุณใส่ค่าตัวเลขลงไป โดยจะแสดงเครื่องหมาย "?" (จากคำสั่ง LDX = $BF) ออกมา เมื่อคุณกด return ตัวอะไรก็ตามที่คุณป้อนเข้าไปจะไปปรากฏอยู่ที่อินพุทบัฟเฟอร์ ($0200)

```
7000-  A2 BF     LDX  #$BF
7002-  20 2E D5  JSR  $D52E
7005-  A9 00     LDA  #$00
7007-  85 B8     STA  $B8
7009-  A9 02     LDA  #$02
700B-  85 B9     STA  $B9
700D-  20 7B DD  JSR  $DD7B
7010-  20 52 E7  JSR  $E752
7013-  A5 51     LDA  $51
7015-  A6 50     LDX  $50
7017-  20 41 F9  JSR  $F941
701A-  20 8B FD  JSR  $FD8B
701D-  4C 00 70  JMP  $7000
```

3

ซับรูทีนที่ $DD7B จะทำหน้าที่อ่านค่าตัวอะไรก็ตามจากตำแหน่งที่กำหนดโดย text pointer ($B8-$B9) ไปเปลี่ยนเป็นค่าตัวเลขที่เครื่องเข้าใจ (ฐานสิบด้วย) เก็บไว้ในหน่วยความจำที่มีชื่อว่า FAC ($9D-$A3)

ซับรูทีนที่ $E752 จะทำหน้าที่เปลี่ยนค่าตัวเลขใน FAC เป็นค่าตัวเลขจำนวนเต็มฐานสิบหกจำนวน 2 ไบท์ เก็บไว้ที่แอดเดรส $50 กับ $51

ซับรูทีนที่ $F941 จะทำหน้าที่พิมพ์ค่าตัวเลขฐานสิบหกที่เก็บอยู่ในรีจิสเตอร์ X ตามด้วยแอคคิวมูเลเตอร์ออกมาในบรรทัดเดียวกัน

ซับรูทีนที่ $FD8B จะทำหน้าที่พิมพ์ return ออกมา

คราวนี้ขอเอาโปรแกรมมาแสดงให้ดูน้อยซักหน่อยนะครับ แล้วที่หายไปอีกอย่างทั้งที่ไม่อยากให้หายไปเลย ก็คือตารางบอก started address ของซับรูทีน หรือที่เรียกเป็นภาษาฝรั่งในหนังสือ "What's Where in the APPLE" by William F. Luebbert ว่า "Atlas" ทั้งนี้ก็เพื่อให้มีเนื้อที่ลงบทความได้มากขึ้น ถ้าคุณต้องการก็ขอให้หาดูจากหนังสือเล่มดังกล่าวได้นะครับ หน้าตาของหนังสือเล่มนี้เป็นยังไงก็ขอให้ดูจากคอลัมน์ที่ชื่อ "ไมโครบุ๊ค" ในไมโครฯ เล่มที่ 1 หนังสือเล่มนี้คุณควรจะมีไว้เลยทีเดียว ถ้าจะเล่นแอสเซมบลี้ให้มันละก้อ เวลาซื้อก็ให้ดีนะครับ เพราะมีหนังสือชื่อเหมือนกันหน้าตาคล้ายกัน แต่เล่มบางกว่า ซึ่งจะไม่ใช่ atlas ที่ผมว่าอยู่ หรือถ้าคุณต้องการแต่ตารางคุณก็ถ่ายเอกสารเฉพาะที่คุณต้องการก็ได้ ไม่เปลืองเงิน